

# XOTcl for OpenACS developers

Martin Matuška

`martin.matuska@wu-wien.ac.at`  
`mm@FreeBSD.org`

Institute for Information Systems and New Media  
Vienna University of Economics and Business Administration

OpenACS/.LRN spring conference 2007  
25.04.2007

## What is this tutorial about?

This tutorial is about to give (at least somewhat satisfactory) answers to the following questions:

- ▶ What is object-oriented programming?
- ▶ What is XOTcl? What can I do with it?
- ▶ What has XOTcl to do with OpenACS?
- ▶ Why should I use XOTcl?
- ▶ How can I use XOTcl with OpenACS?

## Introduction

- Basics of Object-Oriented Programming
- Static vs. Dynamic OOP

## Programming with XOTcl

- XOTcl overview
- Working with XOTcl objects
- Working with XOTcl classes

## OpenACS and XOTcl

- XOTcl vs. OpenACS
- AOLserver module
- Package xotcl-core

# What is object-oriented programming?

Object-oriented programming (OOP) is a computer programming technique that makes use of data structures called **objects** and **classes**.

# Why do we need OOP?

OOP makes developer's life easier! How?

- ▶ code is **reusable** (modules, inheritance)
- ▶ code is **independent** (code replacement, debugging)
- ▶ code is **visualizable** (UML)

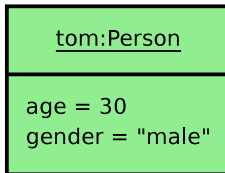
# Objects

Objects are abstract data types that usually represent real entities (e.g. city, train or person)

Objects contain:

- ▶ **attributes** (name, age, ...)
- ▶ **methods** (walk, cry, cook, ...)

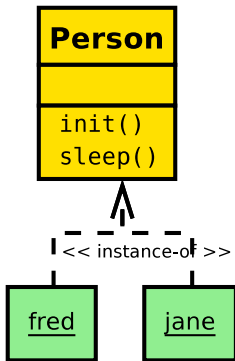
Object attributes are internal object data and can be retrieved and changed only via object methods.



# Classes

Classes are templates for objects. They define attributes and methods for derived objects and classes.

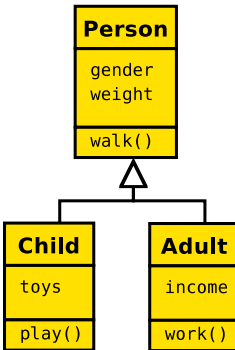
Objects derived from a class are **instances** of that class.



# Inheritance

Classes can build a class hierarchy with superordinated and subordinated classes. Methods are passed down this hierarchy.

- ▶ **simple inheritance** - each class may have only one ancestor
- ▶ **multiple inheritance** - more than one ancestor



# Static vs. Dynamic OOP

## Static OOP:

- ▶ class hierarchy is predefined
- ▶ objects copy classes 1:1
- ▶ examples: C++, Java, Python, PHP, ...

## Dynamic OOP:

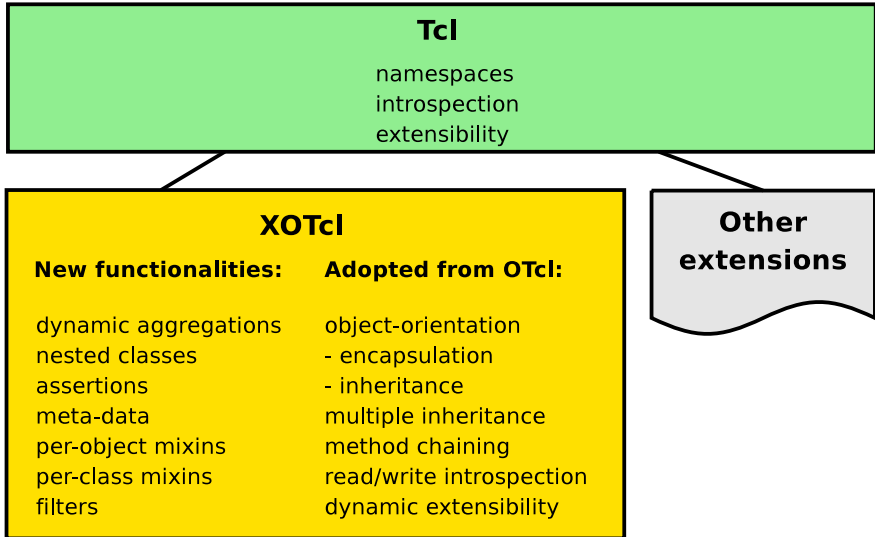
- ▶ class hierarchy can be changed at any time
- ▶ objects are class-independent
- ▶ examples: Smalltalk, CLOS, XOTcl, ...

# What is XOTcl?

Extended Object Tcl (for short: XOTcl, pronounced exotickle) is an dynamic object-oriented scripting language based on Tcl.

Authors: *Gustaf Neumann* and *Uwe Zdun*

- ▶ based on MIT Object Tcl (OTcl)
- ▶ open-source (BSD-style license)
- ▶ included in many Tcl distributions (ActiveTcl, WinTclTk, ...)
- ▶ packaged in several OS'es (Debian Linux, FreeBSD, ...)
- ▶ Tcl binary package "package require XOTcl"



Loading XOTcl and initializing the namespace:

```
% package require XOTcl  
1.5.3  
% namespace import ::xotcl::*
```

First empty object:

```
% Object tom  
::tom  
% Object create tom  
::tom  
% tom destroy
```

Each object registers a new command (*tom*) and is inherited from the meta-class *Object*. Procs provided by *Object* are common to **all** objects and classes.

## Defining object attributes:

```
% tom set friends peter
peter
% tom unset friends
```

## Defining object methods:

```
% tom proc invite {friend} {
    my instvar invited
    lappend invited $friend
    puts "Friends invited: $invited"
}
% tom invite susan
Friends invited: susan
% tom invite peter
Friends invited: susan peter
```

## Object introspection:

```
% tom info vars
actions invited friends
% tom info procs
invite
% tom info class
::xotcl::Object
% tom info args invite
friend
% tom info children
::tom::john
```

## Creating classes:

```
% Class Person
::Person
% Person fred
::fred
% fred info class
::Person
```

## Registering procedures for classes:

```
% Person instproc sleep args {
puts "[self] is sleeping."
}
% fred sleep
::fred is sleeping.
```

## Class hierarchy:

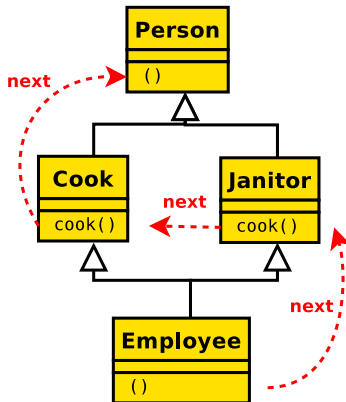
```
% Class Person
::Person
% Class Janitor -superclass Person
::Janitor
% Janitor instproc cook args {
    puts "I have no idea of cooking."
}
% Class Cook -superclass Person
::Cook
% Cook instproc cook args {
puts "I am cooking a meal."
}
```

## Multiple inheritance:

```
% Class Employee -superclass {Janitor Cook}
::Employee
% Employee jerry
::jerry
% jerry cook
I have no idea of cooking.
% Janitor instproc cook args {
    puts "I have no idea of cooking."
    next
}
% jerry cook
I have no idea of cooking.
I am cooking a meal.
```

Precedence order of *next*:

Next invokes a identically named method from the next class (or object) in the **next** precedence list. This order can be retrieved with the **info precedence** method.



## Changing of classes and superclasses:

Object may change their class and classes may change their superclasses on runtime in XOTcl.

```
% Class Male
::Male
% Class Female
::Female
% Male steve
::steve
% steve info class
::Male
% steve class Female
% steve info class
::Female
```

## Class introspection:

```
% Class City
::City
% City paris
::paris
% paris isclass
0
% paris isobject
1
% City isclass
1
% City isobject
1
```

# What has XOTcl to do with OpenACS?

OpenACS/.LRN includes support for XOTcl. There are more packages using XOTcl, here are some examples:

- ▶ **XoWiki** - a wiki implementation for OpenACS in XoTcl
- ▶ **XOTcl Request Monitor** - performance summary monitor for OpenACS applications
- ▶ **Chat** - a web-based chat implementation for OpenACS

# Why should I use XOTcl with OpenACS?

Using XOTcl in OpenACS has many advantages over classic procedural programming. Some of these are:

- ▶ object-orientation improves code clarity and debugging
- ▶ modular structure increases source code reusability
- ▶ package **xotcl-core** provides low-level core functionality that significantly eases operation (e.g. with content repository or thread management)

## How can I use XOTcl with OpenACS?

There are three ways of using XOTcl in OpenACS/.LRN:

1. direct package loading (not recommended)
2. using pre-loaded XOTcl from the AOLserver module
3. using functions provided by the OpenACS xotcl-core package

To take advantage of XOTcl features within OpenACS, the following is required:

- ▶ a working OpenACS/.LRN installation
- ▶ XOTcl binary extension
- ▶ AOLserver XOTcl module
- ▶ xotcl-core OpenACS package

# What is an AOLserver module?

AOLserver modules extend the Tcl function set that was provided by AOLServer (`ns*`) in a similar way packages add new Tcl functions.

There are two types of AOLserver modules:

- ▶ binary modules (`nscache`, `nspostgres`, `nsopenssl`)
- ▶ script modules (`http.tcl`, `sendmail.tcl`)

## Why do I need an AOLserver module?

If every script initiated `package require XOTcl`, this would have extremely negative effects on system performance (speed, memory).

The AOLserver XOTcl module (`xotcl.tcl`) loads and initializes the XOTcl package and bundled XOTcl libraries for all AOLserver threads.

**Result:** high speed, low memory usage, no need for `package require`

## AOLserver blueprint

There is a special way how AOLserver treats Tcl script modules.

Commands from all modules are put into a single tcl command set, the AOLserver **blueprint**.

**Problem:** the blueprint is not ordered, XOTcl requires an exact loading order

**Solution:** the XOTcl module changes the loading order and with help of the *xotcl::serializer* package maintains correct loading order.

## The xotcl-core package

The xotcl-core OpenACS package significantly simplifies package programming by providing several API's.

These API's include:

- ▶ XOTcl simple Content repository API
- ▶ Low level db abstraction API
- ▶ Thread management API (and background delivery)
- ▶ Policies API
- ▶ Api-browser API
- ▶ Generic Chat

## Example: content repository operations

```
CrItem instantiate -item_id 6413
6413 append title " - Works perfectly"
6413 save
6413 save_new
6413 delete
```

```
CrItem instantiate -item_id 6417
6417 class ::xowiki::Page
6417 set title "en:[6417 set title]"
6417 save_new
```

Thank you for your attention!

Tutorial document:

<http://www.matuska.org/martin/doc/xotcl-openacs-2007.pdf>